Game Design Document
Active Sport Game
WS19/20

Blade Runner

Group 5
James Li, Zixiang Wang, Liudongnan Yang

Concept

Story

Ch44, a former cleaning robot, has found his true potential as a cyborg ninja cleaning the city in a slightly different fashion. Equipped with a high-tech-blade he will slice through each and every obstacle that encounters him. But he must be cautious since his power supplies are limited and will quickly drain out. Only his blade can charge him with every hit it lands. With that being said. Happy cutting!

Game content

Blade Runner is a traditional endless-runner game. The player's goal is to navigate through the city while sustaining his gradually decreasing power-charge. In order to achieve this, he has multiple different tasks to fulfill.

One being, avoiding collision with obstacles, using the actions:

- Move Left & Right
- Jump

The second being, cutting close objects to charge his power. The amount of charge will depend on the hits one manages to land.

Gameplay

The character can be controlled via PC Input or Microsoft Kinect, a motion-sensing input device.

Character Movement:

- Left & Right = A & D (PC) / Left & Right arm raise (Kinect)
- Sprint = W (PC) / Not featured in Kinect
- Jump = Space (PC) / Jumping with both hands up (Kinect)
-

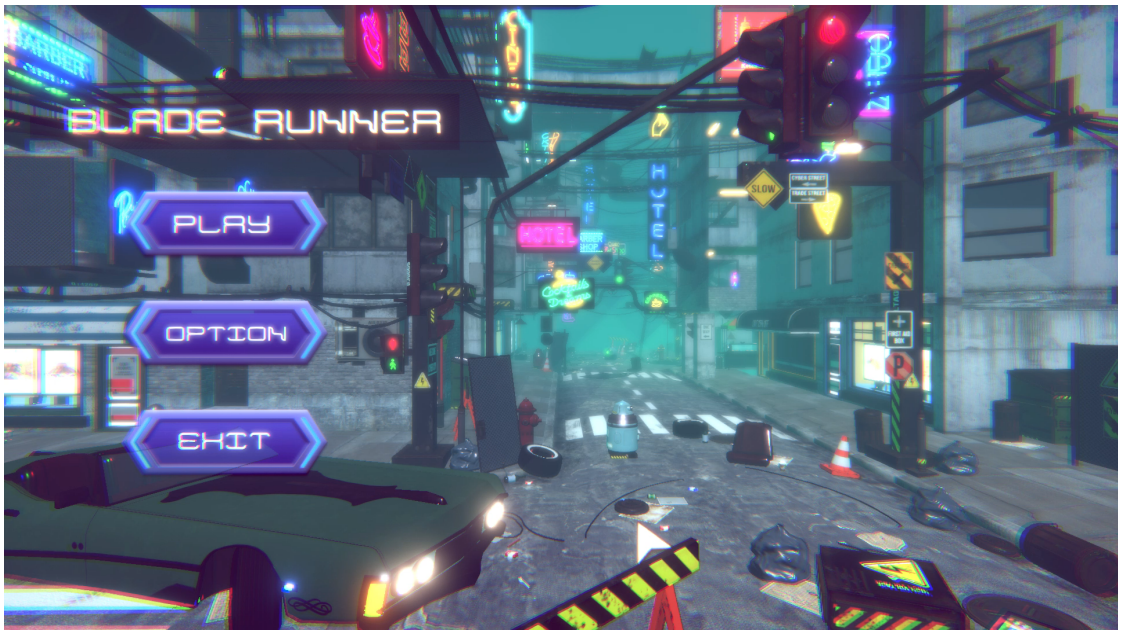
Blade-Mode:

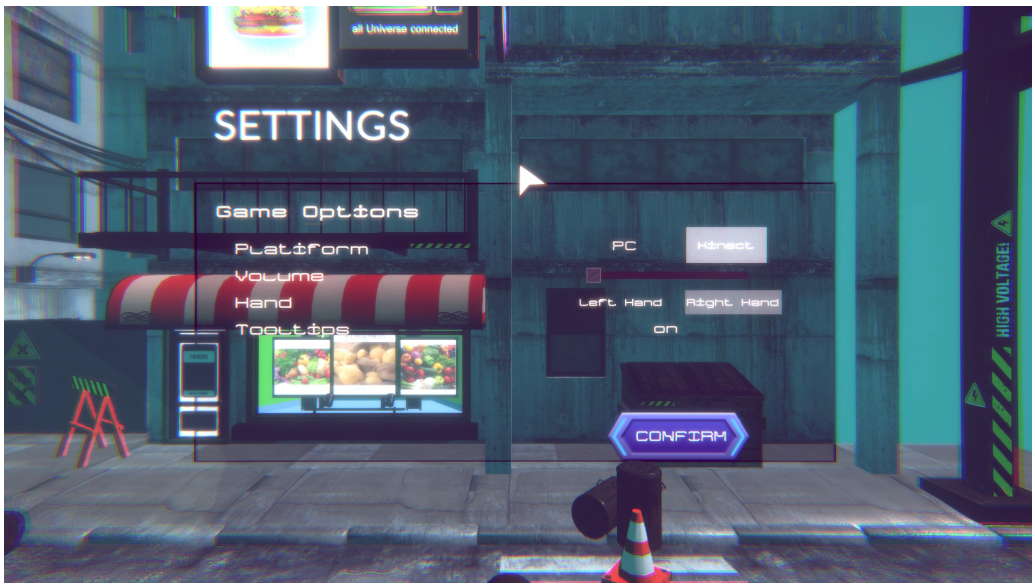
- Activating: Right-Click mouse (PC) / Assemble both hands "Prayer's Pose" (Kinect)
- Slicing: Hold left mouse button & drag into one direction(PC) / Drag hand across the screen (Kinect)

Game Design

The style we focused on is a Neo- SciFi style, where the city resembles “futuristic messiness”. This was achieved by using dense fog and the cyberpunk assets from the unity store.

Title scene





Main scene

The UI elements and animations in the main scene are based on the game, Metal Gear Rising: Revengeance. The yellow bar represents the remaining energy, which will slowly decrease. The red bar on the other hand indicates the blade time and charges over time. Upon activating the Blade-mode, the red bar will act as a time frame, where slicing is enabled.

Procedure

1. Implementation of all the main features to build the game loop for the running system and cutting system.
2. GUI design and scene management during the process between different departments.
3. Sound & visual effect implementation
4. Optimization

Backlog

The development process lasted about 2 months with a team consisting of 3 engineers, James Li, Liudongnan Yang, and Zixiang Wang.

Affairs Table	Due Date	Commit Date
Initialization	Dec.04.2019	Dec.15.2019
GameLoop	Dec.19.2019	Dec.24.2019
Cutting Feature	Dec.15.2019	Dec.24.2019
UI	Jan.5.2020	Jan.10.2020
VFX	Dec.04.2019	Jan.30.2020
Trailer	Feb.01.2020	Feb.05.2020

The whole project process backed up:
https://tum-games-engineering.monday.com/boards/393289691/

https://github.com/hakrrr/Blade-Runner

Technical

A video documentation of the procedure can be found under
https://www.instagram.com/p/B8KM7s1Ba8O/

Slicing feature

For this feature we used an open source framework, Ezy Slice (https://github.com/DavidArayan/ezy-slice) which allowed us to integrate mesh cutting into our scene. The algorithm instantiates 2 new gameobjects by retrieving information from the cutting plane and generating new vertices in respect to the edges.

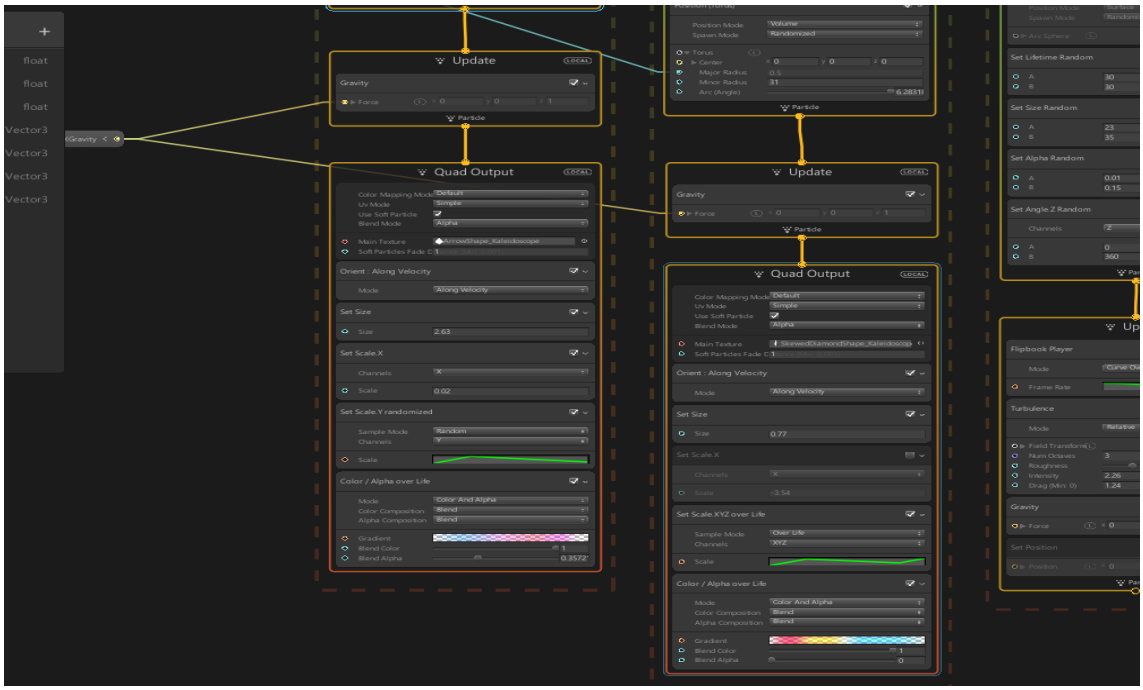
Gesture recognition

Gesture recognition was implemented by using Microsoft's Visual Gesture Builder. Gestures were divided into discrete and continuous ones. For both types the machine required labelled training-data in the form of video recordings. After the training is done, a gdb file (database) will be generated, which then can be accessed in the source code.

VFX

The visual effect leads the project to a new level of quality, therefore after we build up the whole game loop, the effect should have come into the stage. There are a few visual effects that we are used in the game.

The effect in the main scene contains different types of effects such as particle system, shader graph. Specifically need to be mentioned is that in the second scene we used a VFX Graph and the shader graph



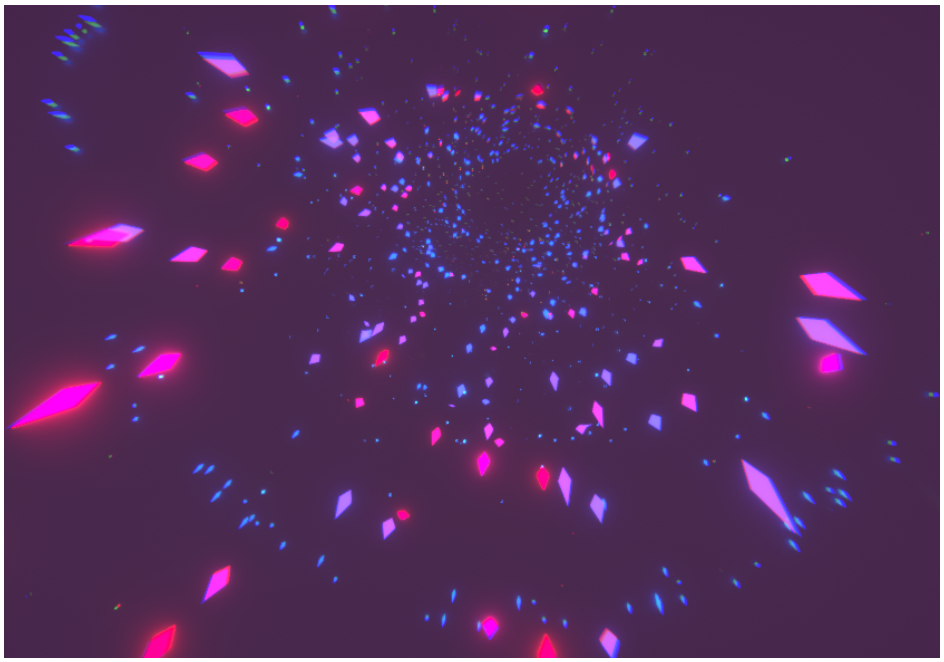
The main idea of the graph is to layer 4 different subgraphs to generate a black hole effect and particles are generated on the ring of Torus.

In the graph, the parameter VFXGravity controls the ring that the effect has made.

The value of VFXGravity will get from the music data processing script: AudioData_Amplitide.cs.In this script, the program will use the microphone component and separated the captured data into 8 bands

1. Sub Bass: 0 - 86Hz
2. Bass: 87 - 258Hz
3. Low-Mids: 259 - 602Hz
4. Mids: 603 - 1290Hz
5. Upper-Mids: 1291 - 2666Hz
6. Presence: 2667 - 5418Hz
7. Brilliance: 6419 - 10922Hz
8. Dog Whistle: 10923 - 21930Hz

After processing the music, we apply the data to the VFX graph and let the frequency band 3 and 8 be our input and send damped value to the parameter VFXGravity by script.



Trailer

The film was shot in Unity using Cinemachine. We set up 4 different tracks for the scene control and 2 for the camera control.